exercice 5, TD 1

question 4 On cherche à calculer la fonction :

$$\phi(x) = \frac{1}{1+x} - \frac{1}{1-x} = \frac{-2x}{(1+x)(1-x)}$$

pour un nombre flottant x tel que $|x|$ est assez petit. Analyser l'erreur obtenue par les deux algorithmes :

1. $y1 := (1 \oslash (1 \oplus x)) \ominus (1 \oslash (1 \ominus x))$
2. $y2 := (-2 \otimes x) \oslash ((1 \oplus x) \otimes (1 \ominus x))$

Commençons par l'algorithme 2, on obtient assez rapidement (il n'y a pas d'erreur avec la multiplication par 2) :

$$y_2 = \phi(x)\frac{1+\epsilon_4}{(1+\epsilon_1)(1+\epsilon_2)(1+\epsilon_3)} = \phi(x)(1+\delta) \quad \text{avec } |\delta| \leq 4\mathbf{u}(/(1-4\mathbf{u}))$$

et donc l'erreur relative est quasi-bornée par $4\mathbf{u}$.

Pour l'algorithme 1, on va avoir une soustraction de deux nombres flottants assez proches et donc calcul exact pour cette soustraction mais amplification des erreurs précédentes :

$$\begin{aligned}
y1 &= (1 \oslash ((1+x)(1+\epsilon_1))) \ominus (1 \oslash ((1-x)(1+\epsilon_2))) \\
&= \frac{1}{(1+x)(1+\epsilon_1)}(1+\epsilon_3) \ominus \frac{1}{(1-x)(1+\epsilon_2)}(1+\epsilon_4) \\
&= \frac{1}{(1+x)(1+\epsilon_1)}(1+\epsilon_3) - \frac{1}{(1-x)(1+\epsilon_2)}(1+\epsilon_4) \quad \text{si } |x| \text{ assez petit} \\
&= \frac{1}{1+x}(1+\delta_1) - \frac{1}{1-x}(1+\delta_2) \quad \text{avec } |\delta_k| \leq 2\mathbf{u}(/(1-2\mathbf{u})) \\
y_1 &= \phi(x) + \frac{\delta_1}{1+x} - \frac{\delta_2}{1-x} \quad = \quad \phi(x)\left(1 + \frac{1}{\phi(x)}\left(\frac{\delta_1}{1+x} - \frac{\delta_2}{1-x}\right)\right)
\end{aligned}$$

En utilisant la deuxième expression de ϕ, l'erreur relative e_r pour cet algorithme peut s'écrire :

$$\begin{aligned}
e_r(x) = \frac{1}{\phi(x)}\left(\frac{\delta_1}{1+x} - \frac{\delta_2}{1-x}\right) &= \frac{-(1-x)\delta_1 + (1+x)\delta_2}{2x} \\
e_r(x) &= \frac{\delta_2 - \delta_1}{2x} + \frac{\delta_1 + \delta_2}{2}
\end{aligned}$$

Ainsi sauf compensation exacte ($\delta_1 = \delta_2$), l'erreur relative va être très grande lorsque $|x|$ devient petit. Il y a donc un avantage net pour le deuxième algorithme lorsque $|x| << 1$. Par exemple prenons $x = 10^{-4}$ et supposons que $\delta_2 = \frac{1}{5}\mathbf{u}$ et $\delta_1 = \frac{1}{10}\mathbf{u}$ alors l'erreur relative est de $1000.15\mathbf{u}$. □

question 5 Même question avec la fonction $f(x) = \sqrt{1+x} - \sqrt{1-x}$. On analysera d'abord l'erreur obtenue par "l'algorithme" immédiat : $y1 := sqrt(1 \oplus x) \ominus sqrt(1 \ominus x)$ où $sqrt(u)$ désigne la racine calculée en machine (pour cette opération la norme IEEE impose aussi que $sqrt(u) = fl(\sqrt{u})$, on a donc $sqrt(u) = \sqrt{u}(1+\epsilon)$ avec $|\epsilon| \leq \mathbf{u}$ pour tout flottant $u \geq 0$ qui n'est pas un nombre spécial (car avec la racine carrée $[\sqrt{\mu}, \sqrt{M}] \subset [m, M]$)). Puis on cherchera à réécrire f différemment pour trouver le bon algorithme (lorsque $|x|$ est petit). Rappel : $\sqrt{1+x} = 1 + x/2 + O(x^2)$.

réponse :

$$\begin{aligned}
y1 &= sqrt(1 \oplus x) \ominus sqrt(1 \ominus x) \\
&= sqrt((1+x)(1+\epsilon_1)) \ominus sqrt((1-x)(1+\epsilon_2)) \\
&= \sqrt{(1+x)(1+\epsilon_1)}(1+\epsilon_3) \ominus \sqrt{(1-x)(1+\epsilon_2)}(1+\epsilon_4) \\
&= \sqrt{(1+x)(1+\epsilon_1)}(1+\epsilon_3) - \sqrt{(1-x)(1+\epsilon_2)}(1+\epsilon_4) \quad \text{si } |x| \text{ assez petit} \\
&= \sqrt{1+x}(1 + \frac{1}{2}\epsilon_1 + O(\epsilon_1^2))(1+\epsilon_3) - \sqrt{1-x}(1 + \frac{1}{2}\epsilon_2 + O(\epsilon_2^2))(1+\epsilon_4) \\
&= \sqrt{1+x}(1+\delta_1) - \sqrt{1-x}(1+\delta_2)
\end{aligned}$$

où les termes δ_k sont "quasi" bornés (en valeur absolue) par $1,5\mathbf{u}$ (comme ϵ_1 est petit devant 1, $1+\frac{1}{2}\epsilon_1$ est une très bonne approximation de $\sqrt{1+\epsilon_1}$).

On obtient donc :

$$y_1 = f(x)(1+\frac{\delta_1\sqrt{1+x}-\delta_2\sqrt{1-x}}{f(x)})$$

En utilisant le développement limité $\sqrt{1+x}=1+x/2+O(x^2)$, on voit que le terme dominant de l'erreur relative va être :

$$er \simeq \frac{\delta_1-\delta_2}{x}$$

et donc de nouveau, on a une forte amplification des erreurs pour $|x| << 1$ (sauf lorsque $\delta_1=\delta_2$).

Lorsque $|x| << 1$, un algorithme stable est obtenu en utilisant l'écriture suivante pour f :

$$f(x)=\sqrt{1+x}-\sqrt{1-x}=(\sqrt{1+x}-\sqrt{1-x})\frac{\sqrt{1+x}+\sqrt{1-x}}{\sqrt{1+x}+\sqrt{1-x}}=\frac{2x}{\sqrt{1+x}+\sqrt{1-x}}$$

exercice 7, TD 1 Parfois au cours d'un calcul, la magnitude d'une quantité intermédiaire q peut ne pas tomber dans l'intervalle $[m,M]$ [1] alors que le résultat final pourrait être correctement approché dans le système flottant. On parle dans ce cas d'overflow ou d'underflow parasite, l'exemple le plus simple étant celui du calcul de la norme d'un vecteur (ici en dimension 2 mais c'est encore plus vrai en dimension supérieure) : $\sqrt{x^2+y^2}$.

1. donner des cas d'overflow et d'underflow parasites pour les flottants 4 octets ;

   réponse : Exemple d'overflow, en prenant $x=y=10^{20}$ le passage au carré de x et y provoque un overflow ($M\simeq 10^{38}$ en "float") alors que dans ce cas le résultat cherché $\sqrt{2}10^{20}$ est tout à fait codable en "float".
   Pour un exemple d'underflow, en prenant $x=y=10^{-23}$ (dans $\mathbb{F}(2,24,-126,127)$ ie dans les "float" on a $m\simeq 1.1710^{-38}$ et $\mu\simeq 1.410^{-45}$) on obtiendra 0. □

2. trouver un algorithme simple qui permet d'éviter ce problème puis obtenir une majoration de l'erreur relative de ce dernier.

   réponse : L'algorithme le plus simple pour éviter ce problème est :

   $X \leftarrow |x|;\ Y \leftarrow |y|$
   si $X<Y$ **alors**
   $\quad temp \leftarrow X;\ X \leftarrow Y;\ Y \leftarrow temp$
   si $X=0$ **alors**
   $\quad$ **retourner** 0
   sinon
   $\quad$ **retourner** $X \times sqrt(1+(Y/X)^2)$

   Pour obtenir une majoration de l'erreur, plaçons nous dans le cas où $|x|\geq|y|$. La formule s'écrit alors :

   $$res = |x| \otimes sqrt(1 \oplus (|y| \oslash |x|)^{\square 2})$$

   où on note $^{\square 2}$ la puissance 2 en machine. On a :

   $$\begin{aligned}
   |y| \oslash |x| &= (|y/x|)(1+\epsilon_1)\\
   (|y| \oslash |x|)^{\square 2} &= (y/x)^2(1+\epsilon_1)^2(1+\epsilon_2)\\
   1 \oplus (|y| \oslash |x|)^{\square 2} &= (1+(y/x)^2(1+\epsilon_1)^2(1+\epsilon_2))(1+\epsilon_3)\\
   sqrt(1 \oplus (|y| \oslash |x|)^{\square 2}) &= \sqrt{(1+(y/x)^2(1+\epsilon_1)^2(1+\epsilon_2))(1+\epsilon_3)}(1+\epsilon_4)\\
   res &= |x| \times \sqrt{(1+(y/x)^2(1+\epsilon_1)^2(1+\epsilon_2))(1+\epsilon_3)}(1+\epsilon_4)(1+\epsilon_5)
   \end{aligned}$$

1. Plus précisément si $|q|<m$ avec $fl(q)\neq q$ on obtient alors une perte de précision (erreur relative plus forcément bornée par $\mathbf{u}$) et dans le cas où $|q|\geq \tilde{M}$ (seuil exact d'overflow) on obtient un Inf.

Soit :

$$res = \sqrt{(x^2 + y^2(1+\epsilon_1)^2(1+\epsilon_2))(1+\epsilon_3)}(1+\epsilon_4)(1+\epsilon_5)$$

Pour simplifier on note $1+\delta = (1+\epsilon_1)^2(1+\epsilon_2)$. On a $|\delta|$ "quasi-borné par $3\mathbf{u}$. On remarque aussi que si $y \neq 0$, on a :

$$\frac{y^2}{x^2+y^2} \leq \frac{y^2}{y^2+y^2} = \frac{1}{2}$$

et si $y = 0$ cette majoration est toujours valable. Donc :

$$\begin{aligned} x^2 + y^2(1+\epsilon_1)^2(1+\epsilon_2) &= x^2 + y^2(1+\delta) \\ &= (x^2+y^2)(1+\frac{y^2}{x^2+y^2}\delta) \\ &= (x^2+y^2)(1+\delta') \end{aligned}$$

avec $|\delta'|$ "quasi-borné" par $(3/2)\mathbf{u}$. Il s'ensuit que :

$$\begin{aligned} res &= \sqrt{x^2+y^2}\,\sqrt{(1+\delta')(1+\epsilon_3)}\,(1+\epsilon_4)(1+\epsilon_5) \\ &= \sqrt{x^2+y^2}\,\sqrt{(1+\delta'')}\,(1+\epsilon_4)(1+\epsilon_5) \\ &= \sqrt{x^2+y^2}\,(1+\delta''')\,(1+\epsilon_4)(1+\epsilon_5) \\ &= \sqrt{x^2+y^2}\,(1+\Delta) \end{aligned}$$

où $|\delta''|$ est "quasi-borné" par $(5/2)\mathbf{u}$, $\delta''' \simeq \delta''/2$ avec une très bonne approximation (δ'' étant très petit par rapport à 1) et donc $|\delta'''|$ est "quasi-borné" par $(5/4)\epsilon_m$. Finalement, $|\Delta|$ l'erreur relative de cette formule sera "quasi-bornée" par : $(3+\frac{1}{4})\mathbf{u}$. □

exercice 9, TD 1 Le but de l'exercice est d'étudier l'algorithme utilisé par la fonction `dnrm2` de la bibliothèque BLAS standard. Cette fonction permet de calculer la norme euclidienne d'un vecteur de $\mathbb{R}^n$:

$$\|x\| = \left(\sum_{k=1}^{n} x_k^2\right)^{1/2} \tag{1}$$

en évitant les problèmes d'overflow parasite.

1. Rappeler ce qu'est un overflow parasite.

   réponse : Un overflow parasite correspond à dépassement de capacité (résultat r d'un calcul [2] tel que $|r| \geq \tilde{M}$) lors d'un calcul intermédiaire alors que le résultat final théorique est codable dans le système flottant utilisé. □

2. Une première version consiste à généraliser la méthode vue en TD pour le cas $n = 2$: on recherche le maximum des $|x_k|$ qui sert alors de facteur d'échelle (on factorise l'expression (1) par $\max_k |x_k|$). Ecrire précisemment cet algorithme.

   réponse :

   $s \leftarrow |x_1|$
   pour k de 2 à n
   　　si $|x_k| > s$ **alors**
   　　　　$s \leftarrow |x_k|$
   $t \leftarrow 0$
   pour k de 1 à n
   　　$t \leftarrow t + (x_k/s)^2$
   retourner $s \times \sqrt{t}$

2. Rappel $\tilde{M}$ nombre très proche de M est le seuil exact d'overflow, cf notes de cours du chapitre 1.

□

3. L'algorithme précédent a le défaut de parcourir deux fois les composantes du vecteur x (une fois pour le max et une deuxième pour le calcul effectif) ce qui n'est pas très efficace du point de vue des accès mémoire. On aimerait faire juste une seule passe sur x ce qui implique l'utilisation d'un facteur d'échelle "dynamique" que l'on modifie au besoin au fur et à mesure des itérations. A la fin de l'itération k le facteur d'échelle doit être égal à :

$$s := \max_{j \in [\![1,k]\!]} |x_j|$$

et on a calculé :

$$t := \sum_{j=1}^{k} \left(\frac{x_j}{s}\right)^2$$

(a) montrer que si $k = n$ on obtient alors $\|x\| = s\sqrt{t}$.

réponse : Il suffit de faire le calcul :

$$s\sqrt{t} = s\left(\sum_{j=1}^{n}\left(\frac{x_j}{s}\right)^2\right)^{1/2} = \left(\sum_{j=1}^{n} s^2\left(\frac{x_j}{s}\right)^2\right)^{1/2} = \left(\sum_{j=1}^{n} x_j^2\right)^{1/2} = \|x\|$$

□

(b) écrire la mise à jour des deux quantités s et t à l'itération $k+1$ (aide : il faut bien sûr tester si $|x_{k+1}| > s$) ;

réponse : si $|x_{k+1}| > s$ il faut changer le facteur d'échelle et en tenir compte aussi dans le calcul de t : il faut multiplier t par s^2 puis diviser par $|x_{k+1}|^2$ mais du fait des risques d'overflow parasite, on multiplie t par $(s/x_{k+1})^2$ (quantité inférieure à 1) ce qui revient au même (mais sans le risque d'overflow parasite). Par la suite on rajoute $|x_{k+1}/x_{k+1}|^2$ (puisque $|x_{k+1}|$ est le nouveau facteur d'échelle) c'est à dire 1.

si $|x_{k+1}| > s$ **alors**
    $t \leftarrow t \times (s/x_{k+1})^2 + 1$
    $s \leftarrow |x_{k+1}|$
sinon
    $t \leftarrow t + (x_{k+1}/s)^2$

(c) écrire alors l'algorithme complet.

$s \leftarrow |x_1|$
$t \leftarrow 1$
pour k de 2 à n
    si $|x_k| > s$ **alors**
        $t \leftarrow t \times (s/x_k)^2 + 1$
        $s \leftarrow |x_k|$
    sinon
        $t \leftarrow t + (x_k/s)^2$
retourner $s \times \sqrt{t}$

□